Scheduling Constrained-Deadline Parallel Tasks on Two-type Heterogeneous Multiprocessors

Björn Andersson
Carnegie Mellon University

Gurulingesh Raravi
Xerox Research Centre India

***Abstract*—Consider the problem of scheduling a taskset on a multiprocessor so that all deadlines are met. Assume (i) constrained-deadline sporadic tasks, i.e., a task generates a sequence of jobs and the deadline of a job is no greater than the minimum inter-arrival time of the task that generates the job, (ii) stage-parallelism, i.e., a task comprises one or more stages with a stage comprising one or many segments so that segments in the same stage are allowed to execute in parallel and a segment is allowed to execute only if all segments of the previous stage have finished, (iii) two-type heterogeneous multiprocessor platform, i.e., there are processors of two types, type-1 and type-2, and for each task, there is a specification of its execution speed on a type-1 processor and on a type-2 processor, and (iv) intra-type migration, i.e., a job can migrate between processors of the same type but for a task, all jobs of this task must execute on the same processor type. We present an algorithm for this problem; it assigns each task to a processor type and then schedules tasks on processors of each type with global-Earliest-Deadline-First. This algorithm has pseudo-polynomial time complexity and speedup factor at most 5. This is the first algorithm for scheduling parallel real-time tasks on a heterogeneous multiprocessor with provably good performance.**

I. Introduction

Software systems are expected to do more with less, i.e., providing more functionality and greater performance with lower size, weight, and power consumption. The real-time systems community has taken a great interest in developing methods that provide foundations for doing so while ensuring, before run-time, that the software system can respond, at run-time, to certain events within pre-specified time constraints (deadlines). Such foundations include algorithms for assigning tasks to heterogeneous multiprocessors. These algorithms are useful because heterogeneous multiprocessors typically provide more processing power per watt. Other foundations include algorithms for scheduling tasks that can execute in parallel on multiprocessors. These algorithms are relevant for computations responding to events that have so tight deadlines that even if a computation is executed on a system with no other computations present, the only way for the computation to meet its deadline is to perform execution in parallel.

A computer platform is a *homogeneous multiprocessor* (sometimes called *identical multiprocessor*) if the execution speed of all tasks is the same on all processors. Conversely, a computer platform is a *heterogeneous multiprocessor* (sometimes called *unrelated multiprocessor*) if the execution speed of a task depends on both the processor and the task. A heterogeneous multiprocessor is *two-type* if it has two types of processors (a.k.a *two-type platform*). Analogously, a heterogeneous multiprocessor is *t-type* if it has t types of processors (a.k.a *t-type platform*). For two-type platforms, the problem of assigning tasks to processors is NP-hard in the strong sense and the problem of assigning tasks to processor types is NP-hard [1]. For t-type platforms, both the problems are NP-Hard in the strong sense [2], [3]. Consequently, the research community has developed approximation algorithms (i.e., algorithms with finite speedup factors) for assigning tasks to processors and to processor types [1]–[15] on such platforms.

Related work. The algorithms for assigning *implicit-deadline* sporadic tasks (i.e., a task generates a sequence of jobs and a job has a deadline that is equal to the minimum inter-arrival time of the task that generates the job) to processors and to processor types for two-type platforms [1], [5]–[8] have lower time complexity than the algorithms for t-type platforms [2]–[4], [9]–[14] while maintaining their performance bound. In addition, an algorithm for scheduling *arbitrary-deadline* sporadic tasks (i.e., a task generates a sequence of jobs and a job has a deadline that may be less than or greater than or equal to the minimum inter-arrival time of the task that generates the job) on t-type platforms is known as well [15]. However, they [1]–[15] do not support parallel tasks. The research community has also presented algorithms with proven speedup factors for scheduling parallel tasks on homogeneous multiprocessors [16]–[22]. Further, there are other algorithms [23]–[33] with *no* proven speedup factors for scheduling parallel tasks on homogeneous multiprocessors — some of them [23]–[28] are for *constrained-deadline* tasks (i.e., a task generates a sequence of jobs and a job has a deadline that may be less than or equal to the minimum inter-arrival time of the task that generates the job) and the others [29]–[33] are for implicit-deadline tasks. Unfortunately, none of these works [16]–[33] support heterogeneous multiprocessors (and moreover most of these algorithms [23]–[33] have no proven speedup factors). A work by Holenderski et. al. [34] comes closest to ours as it also deals with the problem of scheduling parallel tasks on heterogeneous multiprocessors. However, the approach presented in [34] has no proven speedup factor.

This research. In this paper, we present a pseudo-polynomial algorithm for scheduling constrained-deadline parallel tasks on a two-type heterogeneous multiprocessor and prove its speedup factor. Our approach assigns each task to a processor type and then uses global-Earliest-Deadline-First (gEDF) on the processors of each type to schedule the

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
13 JAN 2015	**N/A**	

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
Scheduling Constrained-Deadline Parallel Tasks on Two-type Heterogeneous Multiprocessors	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
Raravi /Bjorn A. Andersson Gurulingesh	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)	8. PERFORMING ORGANIZATION REPORT NUMBER
Software Engineering Institute Carnegie Mellon University Pittsburgh, PA 15213	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release, distribution unlimited.

13. SUPPLEMENTARY NOTES
The original document contains color images.

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			
unclassified	**unclassified**	**unclassified**	**SAR**	**13**	

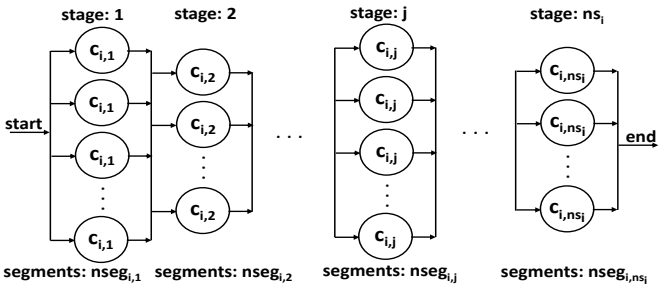

Fig. 1: The parallel task model studied in this paper.

respective tasks. We show that our new algorithm has pseudo-polynomial time complexity and speedup factor at most 5. We study the *constrained-deadline sporadic task model* and we consider *parallelism with stages* (i.e., a task is described with one or more stages with each stage comprising one or many segments such that segments in the same stage are allowed to execute in parallel but a segment is only allowed to execute if all segments of the previous stage have finished execution). This work makes the following contribution: it presents the *first algorithm for scheduling parallel real-time tasks on a heterogeneous multiprocessor with a proven speedup factor*.

Organization of the paper. The rest of this paper is organized as follows. Section II states the system model. Section III lists previous results on parallel scheduling on homogeneous multiprocessors and also proves new lemmas that we use later in the paper. Section IV presents our new algorithm for two-type heterogeneous multiprocessors and proves its speedup factor and time complexity. Section V concludes.

II. System Model

We consider the problem of scheduling a set τ of constrained-deadline sporadic tasks on a two-type heterogeneous multiprocessor platform Π comprising m_1 processors of type-1 and m_2 processors of type-2. A task $\tau_i \in \tau$ is characterized by a *minimum inter-arrival time* T_i and a *deadline* D_i such that $D_i \leq T_i$. Each task τ_i generates a sequence of *jobs*, with the first job arriving at any time and subsequent jobs arriving *at least* T_i time units apart.

The execution of a task τ_i is described by ns_i, $\mathrm{nseg}_{i,j}$, and $C_{i,j}$ with the interpretation that a job of τ_i has ns_i stages with stage j comprising $\mathrm{nseg}_{i,j}$ segments with each segment of stage j having execution requirement at most $C_{i,j}$ — see Fig. 1. A segment finishes when it performs a number of units of execution equal to its execution requirement. A segment executing contiguously for L time units on a processor of speed s performs $L \times s$ units of execution. A segment of a job is allowed to execute only if all segments of its previous stage have finished. A job finishes when all segments of its last stage have finished. If a job of τ_i finishes at most D_i time units after its arrival, then it meets its deadline.

On a two-type platform, the execution speed of a job depends on the type of processor on which it executes. Let r_i^1 and r_i^2 denote the execution speeds of a job of task τ_i when it executes on a processor of type-1 and type-2 respectively.

We now define terms that we use in the rest of the paper.

Definition 1 (**Legal jobset**). *If, for each task in the taskset τ, the task is assigned the number of jobs it generates and each job is assigned an arrival time such that the minimum inter-arrival time constraint is satisfied and each segment of a job is assigned an execution requirement such that the upper bound on execution requirement of a segment is respected, then we say that the resulting jobset is a legal jobset with respect to τ.*

Definition 2 (**Intra-migrative schedule**). *A schedule is intra-migrative if both of the following conditions are true: (i) jobs are allowed to migrate between processors of the same type and (ii) for each task, it holds that, if a job executes on a processor of one type then all other jobs of this task execute on processors of the same type.*

Definition 3 (**Intra-migrative feasible taskset**). *A taskset τ is intra-migrative feasible on a two-type platform Π if for each jobset that is legal with respect to τ there exists an intra-migrative schedule in which all deadlines are met.*

Definition 4 (S**-Schedulable task set**). *A taskset τ is S-schedulable on a two-type platform Π if for each jobset that is legal with respect to τ, for each schedule that S can generate from the jobset, it holds that the schedule is intra-migrative and all deadlines are met.*

Definition 5 (**Speed of the computing platform**). *If Π is a two-type platform then let $\Pi \times x$ denote a two-type platform where the speed of each processor is multiplied by x.*

Definition 6 (**Speedup factor**). *A scheduler S has a speedup factor $\mathrm{SF_S}$ if, for each taskset τ, for each two-type platform Π, it holds that: if τ is intra-migrative feasible on Π then τ is S-schedulable on $\Pi \times \mathrm{SF_S}$.*

In order to simplify our discussion in the rest of the paper, we rewrite our model to an equivalent formulation as follows. Instead of using $C_{i,j}$, r_i^1, and r_i^2, we use parameters $C_{i,j}^1$, $C_{i,j}^2$, and s selected as follows: $C_{i,j}^1/s = C_{i,j}/r_i^1$ and $C_{i,j}^2/s = C_{i,j}/r_i^2$. We let s.t. mean *such that* and : mean *it holds that*. We let $\{x|f(x)\}$ denote a set of elements so that an element x is in the set if and only if $f(x)$ is true. For convenience, we write the predicate $(\forall t > 0 : x)$ to mean the predicate $(\forall t \text{ s.t. } t > 0 : x)$. For convenience, we also define $\mathrm{DMAX} = \max_{\tau_i \in \tau} D_i$, $\mathrm{DMIN} = \min_{\tau_i \in \tau} D_i$, and $\mathrm{TMAX} = \max_{\tau_i \in \tau} T_i$.

III. Schedulability Analysis of Parallel Tasks on a Homogeneous Multiprocessor

It is known that there is no optimal online algorithm for scheduling constrained-deadline sporadic tasks on a homogeneous multiprocessor (even for tasks without parallelism) [35]. Therefore, in this paper, we use global-Earliest-Deadline-First (gEDF) scheduling policy as it has a good speedup factor [36]. There is a brute-force approach [37] which provides exact

$$C_i \stackrel{\text{def}}{=} \sum_{j=1}^{\text{ns}_i} \left(\text{nseg}_{\text{i,j}} \times C_{i,j}\right) \qquad \eta_i \stackrel{\text{def}}{=} \sum_{j=1}^{\text{ns}_i} \left(\left\lceil \frac{\text{nseg}_{\text{i,j}}}{m} \right\rceil \times C_{i,j}\right) \tag{1}$$

$$\text{WJ}\left(\tau_i, t, s\right) \stackrel{\text{def}}{=} \begin{cases} 0 & \text{if t} < 0 \\ \text{WJS}\left(i, t, 1, s\right) & \text{if } 0 \leq \text{t} < \frac{\eta_i}{s} \\ C_i & \text{if } \frac{\eta_i}{s} \leq \text{t} \end{cases} \quad \text{bsp}_{i,j} \stackrel{\text{def}}{=} \frac{C_{i,j}}{s} \times \left\lfloor \frac{\text{nseg}_{i,j}}{m} \right\rfloor \qquad \text{sp}_{i,j} \stackrel{\text{def}}{=} \frac{C_{i,j}}{s} \times \left\lceil \frac{\text{nseg}_{i,j}}{m} \right\rceil \tag{2}$$

$$\text{WJS}\left(i, t, j, s\right) \stackrel{\text{def}}{=} \begin{cases} t \times m \times s & \text{if } 0 \leq t < \text{bsp}_{i,j} \\ \text{bsp}_{i,j} \times m \times s + \left(t - \text{bsp}_{i,j}\right) \times \left(\text{nseg}_{i,j} \bmod m\right) \times s & \text{if } \text{bsp}_{i,j} \leq t < \text{sp}_{i,j} \\ C_{i,j} \times \text{nseg}_{i,j} + \text{WJS}\left(i, t - \text{sp}_{i,j}, j+1, s\right) & \text{if } \text{sp}_{i,j} \leq \text{t} \end{cases} \tag{3}$$

$$\text{ffdbf}\left(\tau_i, t, v, s\right) \stackrel{\text{def}}{=} \left\lfloor \frac{t}{T_i} \right\rfloor \times C_i + C_i - \text{WJ}\left(\tau_i, \left(D_i - \left(t \bmod T_i\right)\right) \times v, s\right) \tag{4}$$

$$\text{h}\left(\tau, m, s, \sigma, t\right) \stackrel{\text{def}}{=} \left(\left(\sum_{\tau_i \in \tau} \text{ffdbf}\left(\tau_i, t, \frac{\sigma}{s}, s\right)\right) \leq \left(\left(m - \left(m-1\right) \times \frac{\sigma}{s}\right) \times t \times s\right)\right) \tag{5}$$

$$\left(\exists \sigma \text{ s.t. } \left(\sigma \geq \max_{\tau_i \in \tau} \frac{\eta_i}{D_i}\right) \wedge \left(\forall t \text{ s.t. } t \geq 0 : \text{h}\left(\tau, m, s, \sigma, t\right)\right)\right) \Rightarrow \tau \text{ is gEDF schedulable on } m \text{ processors of speed } s \tag{6}$$

$$\tau \text{ is feasible on } m \text{ processors of speed } s \Rightarrow \left(\forall t > 0 : \sum_{\tau_i \in \tau} \text{ffdbf}\left(\tau_i, t, 1, s\right) \leq m \times t \times s\right) \tag{7}$$

Fig. 2: Previously known [18] schedulability analysis for gEDF scheduling of parallel tasks on a homogeneous multiprocessor.

$$\text{mi}\left(\tau\right) \stackrel{\text{def}}{=} 2^{\left\lfloor \log_2\left(\max_{\tau_i \in \tau}\left(T_i + D_i\right)\right)\right\rfloor + 1} \tag{8}$$

$$\text{ffdbf}^*\left(\tau_i, t, v, s, \tau\right) \stackrel{\text{def}}{=} \begin{cases} \text{ffdbf}\left(\tau_i, 2^{\lfloor \log_2 t \rfloor + 1}, v, s\right) & \text{if } t \leq \text{mi}\left(\tau\right) \\ \text{ffdbf}\left(\tau_i, \text{mi}\left(\tau\right), v, s\right) + \left(C_i + \frac{C_i}{T_i} \times \left(t - \text{mi}\left(\tau\right)\right)\right) & \text{if } t > \text{mi}\left(\tau\right) \end{cases} \tag{9}$$

Fig. 3: New expressions we will use.

schedulability test for gEDF but it has a very large time-complexity and it does not support parallel tasks and it requires that tasks parameters are integers. Therefore, in this paper, we use a *sufficient* (not exact) schedulability test for parallel tasks.

The research literature provides many sufficient schedulability tests for gEDF for tasks that are not parallel — see for example [38]–[40]. Of particular interest is [40] which provides a schedulability test with a speedup factor of two. This schedulability test states that if there exists a σ such that σ is at least as large as the density of each task and if it holds for each value of t that the sum of ffdbf of tasks is at most a certain value then the taskset is gEDF-schedulable. Here ffdbf means forced-forward demand-bound function; it is a function which describes the maximum amount of execution a given task can demand in a time interval of duration t. Later work [18] extended this for parallel tasks and this was done by defining ffdbf for parallel tasks. Fig. 2 shows this schedulability test (see Eq. (6)) for parallel tasks on a homogeneous multiprocessor [18] comprising m processors. It also shows a feasibility test (see Eq. (7)) for parallel tasks on a homogeneous multiprocessor. Since this formulation is for homogeneous multiprocessors, we do not have the [1] and [2] on $C_{i,j}$. Some basic properties of ffdbf are shown below.

Lemma 1. $\forall t_0 > 0, \forall t > t_0 : \text{ffdbf}\left(\tau_i, t_0, v, s\right) \leq \text{ffdbf}\left(\tau_i, t, v, s\right)$

Proof: Follows from inspection of terms in Eqs. (1)-(4). ■

Lemma 2. $\forall l \in \mathbb{Z} : \text{ffdbf}\left(\tau_i, t + l \times T_i, v, s\right) = \text{ffdbf}\left(\tau_i, t, v, s\right) + l \times C_i$

C1. $\exists \sigma^1$ s.t. $\left(\sigma^1 \geq \delta^{\max,1}(X)\right) \wedge \left(\forall t \geq 0 : \left(\sum_{\tau_i \in \tau} \text{ffdbf}^1\left(\tau_i, t, \frac{\sigma^1}{s}, s\right) \times x_i^1\right) \leq \left(\left(m_1 - (m_1 - 1) \times \frac{\sigma^1}{s}\right) \times t \times s\right)\right)$

C2. $\exists \sigma^2$ s.t. $\left(\sigma^2 \geq \delta^{\max,2}(X)\right) \wedge \left(\forall t \geq 0 : \left(\sum_{\tau_i \in \tau} \text{ffdbf}^2\left(\tau_i, t, \frac{\sigma^2}{s}, s\right) \times x_i^2\right) \leq \left(\left(m_2 - (m_2 - 1) \times \frac{\sigma^2}{s}\right) \times t \times s\right)\right)$

C3. $\forall \tau_i \in \tau : x_i^1 + x_i^2 = 1$

C4. $\forall \tau_i \in \tau : x_i^1 \in \{0, 1\}$ and $x_i^2 \in \{0, 1\}$

Fig. 4: A naive formulation of constraints for task-to-processor-type assignment.

Proof: Follows from Eq. (4). ∎

Let $\text{mi}(\tau)$, in Fig. 3, be a duration of a time interval.

Lemma 3. $\text{ffdbf}(\tau_i, \text{mi}(\tau), v, s) + C_i \leq \text{ffdbf}(\tau_i, 2 \times \text{mi}(\tau), v, s)$

Proof: Applying Lemma 2 with $t = \text{mi}(\tau)$ and $l = 1$, yields $\text{ffdbf}(\tau_i, \text{mi}(\tau), v, s) + C_i = \text{ffdbf}(\tau_i, \text{mi}(\tau) + T_i, v, s)$. From the definition of $\text{mi}(\tau)$, it follows that $T_i \leq \text{mi}(\tau)$. Applying this on the above and using Lemma 1 yields $\text{ffdbf}(\tau_i, \text{mi}(\tau), v, s) + C_i = \text{ffdbf}(\tau_i, 2 \times \text{mi}(\tau), v, s)$. This states the lemma. ∎

We will now introduce a function ffdbf^* (used to over-approximate ffdbf) such that for inputs where t is less than or equal to $\text{mi}(\tau)$, it holds that $\text{ffdbf}^*(\tau_i, t, v, s, \tau)$ is a staircase function and for t greater than $\text{mi}(\tau)$, it holds that $\text{ffdbf}^*(\tau_i, t, v, s, \tau)$ increases linearly with t. Formally, Eq. (9) in Fig. 3 shows the definition of $\text{ffdbf}^*(\tau_i, t, v, s, \tau)$.

Lemma 4. $\forall t > 0 : \text{ffdbf}(\tau_i, t, v, s) \leq \text{ffdbf}^*(\tau_i, t, v, s, \tau)$

Proof: See Appendix. ∎

Definition 7. $TS(\tau, \theta) \stackrel{\text{def}}{=} \{t | (2^{\lfloor \log_2 t \rfloor} = t) \wedge (\text{DMIN} \times (1 - \theta) \leq t \leq \text{mi}(\tau))\} \cup \{2^{\lfloor \log_2(\text{DMIN} \times (1-\theta)) \rfloor}\}$

Lemma 5. $\forall t \in TS(\tau, \theta) : \text{ffdbf}^*(\tau_i, t, v, s, \tau) = \text{ffdbf}(\tau_i, 2 \times t, v, s)$

Proof: Follows from definition of ffdbf^* and Definition 7. ∎

IV. New scheduling algorithm and speedup factor

In this section, we discuss scheduling on a two-type heterogeneous multiprocessor. We will use notations in Fig. 2 and Fig. 3 but with 1 as superscript; this superscript indicates that the quantity if based on $C_{i,j}^1$. Ditto for type-2. For example, from Eq. (1) we obtain: $C_i^1 \stackrel{\text{def}}{=} \sum_{j=1}^{\text{ns}_i} \left(\text{nseg}_{\text{i,j}} \times C_{i,j}^1\right)$ and $C_i^2 \stackrel{\text{def}}{=} \sum_{j=1}^{\text{ns}_i} \left(\text{nseg}_{\text{i,j}} \times C_{i,j}^2\right)$.

A. Developing the new algorithm

The problem of scheduling constrained-deadline parallel sporadic tasks on a two-type heterogeneous multiprocessor. can be solved in two steps. Step 1: Before run-time, assign tasks to processor types so that (i) tasks assigned to type-1 are gEDF-schedulable on the processors of type-1 and (ii) tasks assigned to type-2 are gEDF-schedulable on the processors of type-2. Step 2: At run-time, schedule all tasks assigned to type-1 with gEDF on processors of type-1 and schedule all tasks assigned to type-2 with gEDF on processors of type-2. Since Step 2 is straightforward, we focus our discussion on Step 1.

Step 1 could be solved as follows. Let $x_i^1 = 1$ indicate that task τ_i is assigned to type-1 processors and let $x_i^2 = 1$ indicate that task τ_i is assigned to type-2 processors. Let X denote the matrix of x_i values for all tasks in τ. Then, by using Eq. (6), one could solve Step 1 by assigning values to x_i variables such that all the constraints in Fig. 4 are satisfied. Intuitively, C1 in Fig. 4 states that according to the schedulability test of Eq. (6), the tasks assigned to type-1 processors are gEDF-schedulable on type-1 processors. C2 is analogous for type-2 processors. C3 combined with C4 states that a task is either assigned to type-1 or type-2. C4 states that x_i-variables are integers. Unfortunately, creating an algorithm that assigns values to x_i such that all the constraints in Fig. 4 are satisfied is challenging because (i) it involves an exists-quantifier ($\exists \sigma^1$ in C1 and $\exists \sigma^2$ in C2) and (ii) it involves a forall-quantifier ($\forall t$ in C1 and $\forall t$ in C2) and (iii) it has integer variables. Hence, we will now present other constraints so that if these other constraints are satisfied then the constraints in Fig. 4 are satisfied as well.

Let θ_1 and θ_2 be non-negative parameters that we can choose. Then, instead of asking if there exists a σ^1 in C1 in Fig. 4 with certain properties, let us only consider those σ^1 such that $\sigma^1/s = \theta_1$. Then it follows that if there is a task τ_i with $x_i^1 = 1$ and $\frac{\eta_i^1}{D_i} > \theta_1 \times s(\Pi)$ then C1 is violated. Hence, if θ_1 is given and $\sigma^1/s = \theta_1$ and if $\frac{\eta_i^1}{D_i} > \theta_1 \times s(\Pi)$ then it follows that a necessary condition to satisfy the constraints in Fig. 4 is that $x_i^1 = 0$. We can reason analogously for θ_2 and C2. For this reason, we introduce the following sets.

$$\text{H12} \stackrel{\text{def}}{=} \{\tau_i \in \tau \mid (\frac{\eta_i^1}{D_i} > \theta_1 \times s(\Pi)) \wedge (\frac{\eta_i^2}{D_i} > \theta_2 \times s(\Pi))\} \quad (10)$$

$$\text{H1} \stackrel{\text{def}}{=} \{\tau_i \in \tau \mid (\frac{\eta_i^1}{D_i} \leq \theta_1 \times s(\Pi)) \wedge (\frac{\eta_i^2}{D_i} > \theta_2 \times s(\Pi))\} \quad (11)$$

$$\text{H2} \stackrel{\text{def}}{=} \{\tau_i \in \tau \mid (\frac{\eta_i^1}{D_i} > \theta_1 \times s(\Pi)) \wedge (\frac{\eta_i^2}{D_i} \leq \theta_2 \times s(\Pi))\} \quad (12)$$

$$\text{L} \stackrel{\text{def}}{=} \{\tau_i \in \tau \mid (\frac{\eta_i^1}{D_i} \leq \theta_1 \times s(\Pi)) \wedge (\frac{\eta_i^2}{D_i} \leq \theta_2 \times s(\Pi))\} \quad (13)$$

Observe that $\tau = \text{H12} \cup \text{H1} \cup \text{H2} \cup \text{L}$. We let $\text{H12}(\theta_1, \theta_2, \tau)$ denote H12 for the parameters θ_1, θ_2, τ. Analogously for H1,H2, and L.

Clearly, if θ_1 and θ_2 are given and $\sigma^1/s = \theta_1$ and $\sigma^2/s = \theta_2$ and if there is a task in H12 then it is impossible to satisfy the constraints in Fig. 4. Also, if θ_1 is given and $\sigma^1/s = \theta_1$ then a necessary condition to satisfy the constraints in Fig. 4 is to set, for each task $\tau_i \in \text{H1}$, $x_i^1 = 1$. Analogously, if θ_2 is given and $\sigma^2/s = \theta_2$ then a necessary condition to satisfy the constraints in Fig. 4 is to set, for each task $\tau_i \in \text{H2}$, $x_i^2 = 1$. This gives us the constraints in Fig. 5. It can be seen that if θ_1 and θ_2 are given and for a matrix X, it holds that all

C1. $\forall t \geq 0 : \left(\sum_{\tau_i \in \tau} \text{ffdbf}^1(\tau_i, t, \theta_1, s) \times x_i^1\right) \leq ((m_1 - (m_1 - 1) \times \theta_1) \times t \times s)$
C2. $\forall t \geq 0 : \left(\sum_{\tau_i \in \tau} \text{ffdbf}^2(\tau_i, t, \theta_2, s) \times x_i^2\right) \leq ((m_2 - (m_2 - 1) \times \theta_2) \times t \times s)$
C3. $\forall \tau_i \in \tau : x_i^1 + x_i^2 = 1$
C4. $\forall \tau_i \in \tau : x_i^1 \in \{0, 1\}$ and $x_i^2 \in \{0, 1\}$
C5. $\forall \tau_i \in H1 : x_i^1 = 1$
C6. $\forall \tau_i \in H2 : x_i^2 = 1$
C7. $H12 = \emptyset$

Fig. 5: A slightly less naive formulation of constraints for task-to-processor-type assignment.

C1. $\forall \in TS(\tau, \theta_1) : \left(\sum_{\tau_i \in \tau} \text{ffdbf}^{*1}(\tau_i, t, \theta_1, s, \tau) \times x_i^1\right) \leq ((m_1 - (m_1 - 1) \times \theta_1) \times t \times s)$
C2. $\forall \in TS(\tau, \theta_2) : \left(\sum_{\tau_i \in \tau} \text{ffdbf}^{*2}(\tau_i, t, \theta_2, s, \tau) \times x_i^2\right) \leq ((m_2 - (m_2 - 1) \times \theta_2) \times t \times s)$
C3. $\forall \tau_i \in \tau : x_i^1 + x_i^2 = 1$
C4. $\forall \tau_i \in \tau : x_i^1 \in \{0, 1\}$ and $x_i^2 \in \{0, 1\}$
C5. $\forall \tau_i \in H1 : x_i^1 = 1$
C6. $\forall \tau_i \in H2 : x_i^2 = 1$
C7. $H12 = \emptyset$
C8. $\left(\sum_{\tau_i \in \tau} (C_i^1/T_i) \times x_i^1\right) \leq ((m_1 - (m_1 - 1) \times \theta_1) \times s)$
C9. $\left(\sum_{\tau_i \in \tau} (C_i^2/T_i) \times x_i^2\right) \leq ((m_2 - (m_2 - 1) \times \theta_2) \times s)$

Fig. 6: A formulation of constraints for task-to-processor-type assignment.

C1. $\forall t \in TS(\tau, \theta_1) : \left(\sum_{\tau_i \in \tau} \text{ffdbf}^{*1}(\tau_i, t, \theta_1, s, \tau) \times x_i^1\right) \leq ((m_1 - (m_1 - 1) \times \theta_1) \times t \times s \times 1/2)$
C2. $\forall t \in TS(\tau, \theta_2) : \left(\sum_{\tau_i \in \tau} \text{ffdbf}^{*2}(\tau_i, t, \theta_2, s, \tau) \times x_i^2\right) \leq ((m_2 - (m_2 - 1) \times \theta_2) \times t \times s \times 1/2)$
C3. $\forall \tau_i \in \tau : x_i^1 + x_i^2 = 1$
C4. $\forall \tau_i \in \tau : x_i^1 \geq 0$ and $x_i^2 \geq 0$
C5. $\forall \tau_i \in H1 : x_i^1 = 1$
C6. $\forall \tau_i \in H2 : x_i^2 = 1$
C7. $H12 = \emptyset$
C8. $\left(\sum_{\tau_i \in \tau} (C_i^1/T_i) \times x_i^1\right) \leq ((m_1 - (m_1 - 1) \times \theta_1) \times s \times 1/2)$
C9. $\left(\sum_{\tau_i \in \tau} (C_i^2/T_i) \times x_i^2\right) \leq ((m_2 - (m_2 - 1) \times \theta_2) \times s \times 1/2)$

Fig. 7: A formulation of constraints for task-to-processor-type assignment — relaxed to LP.

C1. $\forall t \in TS(\tau, \theta_1) : \left(\sum_{\tau_i \in \tau} \text{ffdbf}^{*1}(\tau_i, t, \theta_1, s, \tau) \times x_i^1\right) \leq ((m_1 - (m_1 - 1) \times \theta_1) \times t \times s)$
C2. $\forall t \in TS(\tau, \theta_2) : \left(\sum_{\tau_i \in \tau} \text{ffdbf}^{*2}(\tau_i, t, \theta_2, s, \tau) \times x_i^2\right) \leq ((m_2 - (m_2 - 1) \times \theta_2) \times t \times s)$
C3. $\forall \tau_i \in \tau : x_i^1 + x_i^2 = 1$
C4. $\forall \tau_i \in \tau : x_i^1 \in \{0, 1\}$ and $x_i^2 \in \{0, 1\}$
C5. $\forall \tau_i \in H1 : x_i^1 = 1$
C6. $\forall \tau_i \in H2 : x_i^2 = 1$
C7. $H12 = \emptyset$
C8. $\left(\sum_{\tau_i \in \tau} (C_i^1/T_i) \times x_i^1\right) \leq ((m_1 - (m_1 - 1) \times \theta_1) \times s)$
C9. $\left(\sum_{\tau_i \in \tau} (C_i^2/T_i) \times x_i^2\right) \leq ((m_2 - (m_2 - 1) \times \theta_2) \times s)$
C10. $\forall \tau_i \in \tau$ s.t. $\left(\left(x'^1_i = 1\right) \vee \left(x'^2_i = 1\right)\right) : x_i^1 = x'^1_i$
C11. $\forall \tau_i \in \tau$ s.t. $\left(\left(x'^1_i = 1\right) \vee \left(x'^2_i = 1\right)\right) : x_i^2 = x'^2_i$
C12. X' is the solution to the problem in Fig. 7

Fig. 8: A formulation of constraints for task-to-processor-type assignment; we will show that this can be computed in pseudo-polynomial time.

constraints in Fig. 5 are satisfied then all constraints in Fig. 4 are satisfied as well.

Note that there is still a $\forall t$ in C1 and C2 in Fig. 5. We will now present a set of constraints where we only check a finite number of t — see Fig. 6.

Lemma 6. if *X satisfies Fig. 6* **then** *X satisfies Fig. 5.*

Proof: Suppose that the lemma was false. Then there exists $\tau, \Pi, \theta_1, \theta_2, X$ such that X satisfies Fig. 6 and X does not satisfy Fig. 5. Note that it can only be C1 or C2 (or both) in Fig. 5 that are violated.

If it is C1 then we can reason as follows: There must be a

t that violated C1 in Fig. 5. Hence,

$$\left(\sum_{\tau_i \in \tau} \text{ffdbf}^1(\tau_i, t, \theta_1, s) \times x_i^1\right) > (m_1 - (m_1 - 1) \times \theta_1) \times t \times s \quad (14)$$

Let us explore three possibilities:

Case 1: $t > \text{mi}(\tau)$**.** Note that $\text{mi}(\tau)$ is an element in $TS(\tau, \theta_1)$. This gives us from Fig. 6:

$$\left(\sum_{\tau_i \in \tau} \text{ffdbf}^{*1}(\tau_i, \text{mi}(\tau), \theta_1, s, \tau) \times x_i^1\right) \leq (m_1 - (m_1 - 1) \times \theta_1) \times \text{mi}(\tau) \times s \quad (15)$$

Because of C8 in Fig. 6, it also holds that:

$$\left(\sum_{\tau_i \in \tau} (C_i/T_i) \times x_i^1\right) \leq (m_1 - (m_1 - 1) \times \theta_1) \times s \quad (16)$$

Multiplying Eq. (16) by $(t - \text{mi}(\tau))$ and adding to Eq. (15) and then combining with Eq. 14 yields:

$$\left(\sum_{\tau_i \in \tau} (\text{ffdbf}^{*1}(\tau_i, \text{mi}(\tau), \theta_1, s, \tau) + \frac{C_i}{T_i} \times (t - \text{mi}(\tau))) \times x_i^1\right) < \left(\sum_{\tau_i \in \tau} \text{ffdbf}^1(\tau_i, t, \theta_1, s) \times x_i^1\right)$$

Since $TS(\tau, \theta_1)$, we can apply Lemma 5 on the left-most term. Doing so and then applying Lemma 3 yields:

$$\left(\sum_{\tau_i \in \tau} (\text{ffdbf}^1(\tau_i, \text{mi}(\tau), \theta_1, s) + C_i + \frac{C_i}{T_i} \times (t - \text{mi}(\tau))) \times x_i^1\right) < \left(\sum_{\tau_i \in \tau} \text{ffdbf}^1(\tau_i, t, \theta_1, s) \times x_i^1\right)$$

Note that the left-hand side is the expression of ffdbf^{*1} (the second case of Eq. 9). Hence:

$$\sum_{\tau_i \in \tau} \text{ffdbf}^{*1}(\tau_i, t, \theta_1, s, \tau) \times x_i^1 < \sum_{\tau_i \in \tau} \text{ffdbf}^1(\tau_i, t, \theta_1, s) \times x_i^1$$

But this contradicts Lemma 4.

Case 2: $t < \text{DMIN} \times (1 - \theta_1)$**.** For such a t, it holds that $\text{ffdbf}^1(\tau_i, t, \theta_1, s)$ is zero. But this violates Eq. 14.

Case 3: $\text{DMIN} \times (1 - \theta_1) \leq t \leq \text{mi}(\tau)$**.** Let us define t_1 as $t_1 = 2^{\lfloor \log_2 t \rfloor + 1}$ and let t_0 be $t_1/2$. It is easy to see that $t_0 \leq t < t_1$. Note that t_0 is an element in $TS(\tau, \theta_1)$. This gives us from Fig. 6:

$$\left(\sum_{\tau_i \in \tau} \text{ffdbf}^{*1}(\tau_i, t_0, \theta_1, s, \tau) \times x_i^1\right) \leq (m_1 - (m_1 - 1) \times \theta_1) \times t_0 \times s \quad (17)$$

Since $t_0 \leq t$ it clearly holds that:

$$(m_1 - (m_1 - 1) \times \theta_1) \times t_0 \times s \leq (m_1 - (m_1 - 1) \times \theta_1) \times t \times s \quad (18)$$

Lemma 5 yields: $\text{ffdbf}^{*1}(\tau_i, t_0, \theta_1, s, \tau) = \text{ffdbf}^1(\tau_i, t_1, \theta_1, s)$. Combining this with Eq. (14), Eq. (17), and Eq. (18) yields:

$$\sum_{\tau_i \in \tau} \text{ffdbf}^1(\tau_i, t_1, \theta_1, s) \times x_i^1 < \sum_{\tau_i \in \tau} \text{ffdbf}^1(\tau_i, t, \theta_1, s) \times x_i^1$$

Using this and $t < t_1$ and Lemma 1 yields a contradiction.

If C2 is violated then we can reason analogously to the case when C1 is violated.

It can be seen that if the lemma is false then each case results in contradiction. Hence, the lemma is true. ■

Note that in Fig. 6, there is a finite number of constraints and this is what we want. However, the X variables are integers and this makes the problem a Mixed-Integer Linear Program (MILP); the research literature currently neither offers a polynomial time algorithm for solving general MILP nor for solving MILP with the special structure of Fig. 6. For Linear Programming (LP), polynomial time algorithms are known though (see [41], for example). We will now discuss how to exploit this. Fig. 7 shows an LP; it differs from Fig. 6 in that X variables are real numbers instead of integers and it is also more constrained — $s/2$ instead of s in C1,C2,C8,C9. With the solution to this LP, we obtain a new optimization problem — see Fig. 8. This optimization problem is as follows. First, we solve the LP (specified by Fig. 7) and obtain a solution X'. With this solution X', we consider the MILP (specified by Fig. 6) and require that for those i such that ${x'}_i^1 = 1$ or ${x'}_i^2 = 1$, the value of x_i^1 should be equal to ${x'}_i^1$ and the value of x_i^2 should be equal to ${x'}_i^2$. There may be some i:s that remain; these will be assigned values by solving a MILP (as specified by Fig. 8).

Lemma 7. If *θ_1 and θ_2 are given and X satisfies Fig. 8* **then** *X satisfies Fig. 4.*

Proof: Follows from the discussion in this subsection. ■

Hence, solving Fig. 8 yields an assignment of tasks to processor types.

B. Stating the new algorithm

We let $\text{solvePTMILP}(\tau, \Pi, \theta_1, \theta_2)$ denote a function which takes as input a taskset τ and a computer platform Π and θ_1 and θ_2 and returns a tuple $\langle f, X \rangle$ where f is a boolean and X is a matrix with the following interpretation: if Fig. 8 is feasible then f is true and X is the solution; if Fig. 8 is infeasible then f is false and X is undefined.

Algorithm 1 lists the pseudo-code for evaluating the function $\text{solvePTMILP}(\tau, \Pi, \theta_1, \theta_2)$.

Definition 8.

$$R(\Pi) = 4 + \max\left(1 - \frac{1}{m_1}, 1 - \frac{1}{m_2}\right)$$

Algorithm 2 shows how the assignment of tasks to processor types works.

Theorem 1. If *$\langle f, X \rangle = \text{solvePTMILP}(\tau, \Pi, \theta_1, \theta_2)$ and f is true and tasks are assigned to processor types according*

Algorithm 1: An algorithm for evaluating the function solvePTMILP$(\tau, \Pi, \theta_1, \theta_2)$.

Input : A taskset τ and a two-type platform Π and θ_1 and θ_2
Output: A tuple $\langle f, X \rangle$ where f is a boolean and X is a matrix

1 **if** $H12 = \emptyset$ **then**
2 Solve the LP in Fig. 7 and obtain a vertex solution
3 **if** *this LP is feasible* **then**
4 Let X' denote this solution.
5 Let F denote a set of indices of tasks in L such that $(x'^1_i \neq 1) \vee (x'^2_i \neq 1)$.
6 Let us introduce χ^{found} which is an assignment of values to the x_i-variables whose subscript index is in F; this assignment is initialized to be undefined.
7 Let us introduce a local variable foundPTMILP that is boolean and initialize it to false.
8 **foreach** *assignment of 0-1 to the x_i-variables whose subscript index is in F* **do**
9 Evaluate if Fig. 8 is satisfied for this assignment
10 **if** *the above evaluation yields true* **then**
11 Let χ denote the assignment of 0-1 to the x_i-variables whose subscript index is in F
12 **if** foundPTMILP $= false$ **then**
13 Set foundPTMILP to true
14 Set χ^{found} to χ
15 **end**
16 **end**
17 **end**
18 **if** foundPTMILP **then**
19 Form the matrix X as follows:
20 For each $i \in F$: Assign x^1_i and x^2_i according to χ^{found}.
21 For each $i \in L \setminus F$: Assign x^1_i and x^2_i according to X'.
22 For each $i \in H1$: Assign $x^1_i = 1$ and $x^2_i = 0$
23 For each $i \in H2$: Assign $x^1_i = 0$ and $x^2_i = 1$
24 return $\langle \text{true}, X \rangle$
25 **else**
26 return $\langle \text{false}, X' \rangle$
27 **end**
28 **else**
29 return $\langle \text{false}, X \rangle$, where X is undefined.
30 **end**
31 **else**
32 return $\langle \text{false}, X \rangle$, where X is undefined.
33 **end**

Algorithm 2: The new intra-migrative task assignment algorithm for two-type heterogeneous multiprocessors.

Input : A taskset τ and a two-type platform Π
Output: An assignment of tasks to processor types indicated by matrix X

1 $\langle f, X \rangle$:= solvePTMILP$(\tau, \Pi, 1/R(\Pi), 1/R(\Pi))$
2 **if** $(f = \text{true})$ **then**
3 declare SUCCESS and stop
4 **else**
5 declare FAILURE and stop
6 **end**

to the X-matrix and tasks are scheduled with gEDF on each processor type **then** *all deadlines will be met at run-time.*

Proof: Follows from Lemma 7 and the fact that Eq. (6) is a schedulability test. ∎

Theorem 2. If *Algorithm 2 declares success and tasks are assigned to processor types according to the X-matrix and tasks are scheduled with gEDF on each processor type* **then** *all deadlines will be met at run-time.*

Proof: Follows from Theorem 1. ∎

C. Proving the time complexity of the new algorithm

Lemma 8. $|\text{TS}(\tau, \theta_1)| = \lfloor \log_2 \frac{\text{mi}(\tau)}{\text{DMIN} \times (1-\theta_1)} \rfloor + 2$ *and* $|\text{TS}(\tau, \theta_2)| = \lfloor \log_2 \frac{\text{mi}(\tau)}{\text{DMIN} \times (1-\theta_2)} \rfloor + 2$

Proof: Follows from the definition of TS — see Definition 7. ∎

Lemma 9. *After line 5 of Algorithm 1 has executed, it holds that:* $|F| \leq \lfloor \log_2 \frac{\text{mi}(\tau)}{\text{DMIN} \times (1-\theta_1)} \rfloor + \lfloor \log_2 \frac{\text{mi}(\tau)}{\text{DMIN} \times (1-\theta_2)} \rfloor + 6$.

Proof: In the LP, solved on line 2 of Algorithm 1, there are $2 \times L$ variables and there are $\lfloor \log_2 \frac{\text{mi}(\tau)}{\text{DMIN} \times (1-\theta_1)} \rfloor + \lfloor \log_2 \frac{\text{mi}(\tau)}{\text{DMIN} \times (1-\theta_2)} \rfloor + 6 + |L|$ constraints ($\lfloor \log_2 \frac{\text{mi}(\tau)}{\text{DMIN} \times (1-\theta_1)} \rfloor + 2$ constraints of C1; the same number of C2; one constraint of C8; one constraint of C9; $|L|$ constraints of C3). Let X' denote the solution of the LP in Fig. 7. Since for each vertex solution of LP, it holds that the number of non-zero variables is at most the number of constraints, it follows that there are at most $\lfloor \log_2 \frac{\text{mi}(\tau)}{\text{DMIN} \times (1-\theta_1)} \rfloor + \lfloor \log_2 \frac{\text{mi}(\tau)}{\text{DMIN} \times (1-\theta_2)} \rfloor + 6 + |L|$ non-zero values of the X' variables. For each task in $L \setminus F$, it holds that there is exactly one non-zero value in X'. For each task in F, it holds that there is exactly two non-zero values in X'. Hence, there are $1 \times (|L| - |F|) + 2 \times |F|$ non-zero values of the X' variables. Consequently: $1 \times (|L| - |F|) + 2 \times |F| \leq \lfloor \log_2 \frac{\text{mi}(\tau)}{\text{DMIN} \times (1-\theta_1)} \rfloor + \lfloor \log_2 \frac{\text{mi}(\tau)}{\text{DMIN} \times (1-\theta_2)} \rfloor + 6 + |L|$. Rewriting yields: $|F| \leq \lfloor \log_2 \frac{\text{mi}(\tau)}{\text{DMIN} \times (1-\theta_1)} \rfloor + \lfloor \log_2 \frac{\text{mi}(\tau)}{\text{DMIN} \times (1-\theta_2)} \rfloor + 6$. This states the lemma. ∎

Lemma 10. *The number of iterations of the for-loop on line 8 of Algorithm 1 is at most:* $\left(\frac{\text{TMAX}+\text{DMAX}}{\text{DMIN}}\right)^2 \times \frac{256}{(1-\theta_1) \times (1-\theta_2)}$

Proof: The number of iterations is at most $2^{|F|}$. Using Lemma 9 yields that the number of iterations is at most:

$$2^{\lfloor \log_2 \frac{\text{mi}(\tau)}{\text{DMIN} \times (1-\theta_1)} \rfloor + \lfloor \log_2 \frac{\text{mi}(\tau)}{\text{DMIN} \times (1-\theta_2)} \rfloor + 6}$$

Observing that $\text{mi}(\tau) \leq 2 \times (\text{TMAX} + \text{DMAX})$ and rewriting yields the lemma. ∎

Lemma 11. *The time complexity of Algorithm 1 is* $O\left(poly + \left(\frac{\text{TMAX}+\text{DMAX}}{\text{DMIN}}\right)^2 \times \frac{1}{(1-\theta_1) \times (1-\theta_2)}\right)$.

Proof: Follows from the facts that (i) linear programs can be solved in polynomial time [41] and hence line 2 of Algorithm 1 can be performed in polynomial time and (ii) the for number of combinations iterated through in the for-loop of line 8 is at most $\left(\frac{\text{TMAX}+\text{DMAX}}{\text{DMIN}}\right)^2 \times \frac{256}{(1-\theta_1) \times (1-\theta_2)}$. (Follows from Lemma 10.) ∎

Theorem 3. *The time complexity of Algorithm 2 is* $O\left(poly + \left(\frac{\text{TMAX}+\text{DMAX}}{\text{DMIN}}\right)^2\right)$.

Proof: Since Algorithm 2 calls solvePTMILP once with $\theta_1 = \theta_2 = 1/R(\Pi)$ it follows that (using Lemma 11) the time complexity of Algorithm 2 is

$O\left(poly + \left(\frac{\text{TMAX+DMAX}}{\text{DMIN}}\right)^2 \times \frac{1}{(1-1/R(\Pi))\times(1-1/R(\Pi))}\right)$. Observing that $4 \leq R(\Pi)$ yields that the time complexity of Algorithm 2 is $O\left(poly + \left(\frac{\text{TMAX+DMAX}}{\text{DMIN}}\right)^2\right)$. ■

D. Proving the speedup factor of the new algorithm

We will start by discussing necessary conditions for intra-migrative feasibility and then to prove the speedup factor.

Lemma 12. *Consider a taskset τ and a computer platform Π.* **If** *τ is intra-migrative feasible on Π* **then** *there exists a matrix X such that all constraints in Fig. 9 are satisfied.*

Proof: Follows from the fact that Eq. (7) is a necessary condition for feasibility. ■

Lemma 13. *Consider a taskset τ and a computer platform Π.* **If** *τ is intra-migrative feasible on $\Pi\times 1/R(\Pi)$* **then** *there exists a matrix X such that all constraints in Fig. 10 are satisfied.*

Proof: Follows from applying Lemma 12 on $\Pi \times (1/R(\Pi))$ and then considering $t \to \infty$ on C1 and C2 yields C8 and C9 respectively. ■

Lemma 14. $\forall Q \geq 0:$
$(\forall t \geq 0 : \left(\sum_{\tau_i \in \tau} \text{ffdbf}^1(\tau_i, t, \theta_1, s)\right) \leq m \times t \times Q)$
$\Rightarrow$
$(\forall t \in TS(\tau, \theta_1) : \left(\sum_{\tau_i \in \tau} \text{ffdbf}^{*1}(\tau_i, t, \theta_1, s, \tau)\right) \leq m \times t \times Q \times 2)$

Proof: See Appendix. ■

Lemma 15. $\forall Q \geq 0:$
$(\forall t \geq 0 : \left(\sum_{\tau_i \in \tau} \text{ffdbf}^2(\tau_i, t, \theta_2, s)\right) \leq m \times t \times Q)$
$\Rightarrow$
$(\forall t \in TS(\tau, \theta_2) : \left(\sum_{\tau_i \in \tau} \text{ffdbf}^{*2}(\tau_i, t, \theta_2, s, \tau)\right) \leq m \times t \times Q \times 2)$

Proof: See Appendix. ■

Lemma 16. *Consider a taskset τ and a computer platform Π.* **If** *X satisfies Fig. 10* **then** *X satisfies Fig. 11.*

Proof: Follows from applying Lemma 14 on C1 in Fig. 10 and applying Lemma 15 on C2 in Fig. 10. ■

Lemma 17. *Consider a taskset τ and a computer platform Π.* **If** *τ is intra-migrative feasible on $\Pi\times 1/R(\Pi)$* **then** *there exists a matrix X such that all constraints in Fig. 11 are satisfied.*

Proof: Follows Lemma 13 and Lemma 16. ■

Lemma 18. *Consider a taskset τ and a computer platform Π.* **If** *τ is intra-migrative feasible on $\Pi\times 1/R(\Pi)$* **then** *there exists a matrix X such that all constraints in Fig. 12 are satisfied.*

Proof: Algebraic manipulations $of R(\Pi)$ (from Definition 8) yields:

$$m_1 \times t \times \frac{s}{R(\Pi)} \times 2 \leq (m_1 - \frac{m_1 - 1}{R(\Pi)}) \times t \times \frac{s}{2} \quad (19)$$

$$m_2 \times t \times \frac{s}{R(\Pi)} \times 2 \leq (m_2 - \frac{m_2 - 1}{R(\Pi)}) \times t \times \frac{s}{2} \quad (20)$$

$$m_1 \times \frac{s}{R(\Pi)} \leq (m_1 - \frac{m_1 - 1}{R(\Pi)}) \times \frac{s}{2} \quad (21)$$

$$m_2 \times \frac{s}{R(\Pi)} \leq (m_2 - \frac{m_2 - 1}{R(\Pi)}) \times \frac{s}{2} \quad (22)$$

Hence, if X satisfies Fig. 11 then it also satisfies Fig. 12. Combining this with Lemma 17 yields the lemma. ■

Lemma 19. *Consider a taskset τ and a computer platform Π.* **If** *there exists a matrix X such that all constraints in Fig. 12 are satisfied* **then** *Algorithm 2 declares SUCCESS.*

Proof: Let us suppose that the lemma was false. Then there is a taskset τ and a computer platform Π such that

$$\text{there exists a matrix } X \text{ such that all constraints in Fig. 12 are satisfied} \quad (23)$$

and Algorithm 2 declares FAILURE.

Relaxing C4 in Fig. 12 yields:

$$\text{there exists a matrix } X \text{ such that all constraints in Fig. 13 are satisfied} \quad (24)$$

Eq. (23) yields

$$\text{H12}(1/R(\Pi), 1/R(\Pi), \tau) = \emptyset \quad (25)$$

Consider Fig. 7 with $\theta_1 = \theta_2 = 1/R(\Pi)$ and compare with Fig. 13. They are identical. Hence:

$$\text{there is a matrix } X \text{ such that all constraints in Fig. 7 are satisfied for } \theta_1 = \theta_2 = 1/R(\Pi) \quad (26)$$

From the statement that Algorithm 2 declares FAILURE it follows that Algorithm 1 fails for the input $\tau, \Pi, 1/R(\Pi), 1/R(\Pi)$. Since it fails, let us explore the possible lines at which it can fail.

Case 1. Algorithm declares FAILURE **on line 32.** Te condition of the case yields $\text{H12}(1/R(\Pi), 1/R(\Pi), \tau) \neq \emptyset$. But this contradicts Eq. (25).

Case 2. Algorithm declares FAILURE **on line 29.** From the condition of the case, it follows that

$$\text{there exists no matrix } X \text{ such that all constraints in Fig. 7 are satisfied for } \theta_1 = \theta_2 = 1/R(\Pi)$$

But this contradicts Eq. (26).

Case 3. Algorithm declares FAILURE **on line 26.** From the condition of the case, it follows that foundPTMILP is false when the algorithm declares FAILURE on line 26. Let us partition τ into F and $\tau \setminus F$. Note that for $\tau \setminus F$ it holds that X' satisfies Fig. 7 and since this set of tasks have x_i^1 and x_i^2

C1. $\forall t \geq 0 : \left(\sum_{\tau_i \in \tau} \text{ffdbf}^1(\tau_i, t, 1, s) \times x_i^1\right) \leq m_1 \times t \times s$
C2. $\forall t \geq 0 : \left(\sum_{\tau_i \in \tau} \text{ffdbf}^2(\tau_i, t, 1, s) \times x_i^2\right) \leq m_2 \times t \times s$
C3. $\forall \tau_i \in \tau : x_i^1 + x_i^2 = 1$
C4. $\forall \tau_i \in \tau : x_i^1 \in \{0, 1\}$ and $x_i^2 \in \{0, 1\}$
C5. $\forall \tau_i \in \text{H1}(1, 1, \tau) : x_i^1 = 1$
C6. $\forall \tau_i \in \text{H2}(1, 1, \tau) : x_i^2 = 1$
C7. $\text{H12}(1, 1, \tau) = \emptyset$

Fig. 9: Constraints expressing a necessary intra-migrative feasibility condition.

C1. $\forall t \geq 0 : \left(\sum_{\tau_i \in \tau} \text{ffdbf}^1(\tau_i, t, 1/R(\Pi), s) \times x_i^1\right) \leq m_1 \times t \times (s/R(\Pi))$
C2. $\forall t \geq 0 : \left(\sum_{\tau_i \in \tau} \text{ffdbf}^2(\tau_i, t, 1/R(\Pi), s) \times x_i^2\right) \leq m_2 \times t \times (s/R(\Pi))$
C3. $\forall \tau_i \in \tau : x_i^1 + x_i^2 = 1$
C4. $\forall \tau_i \in \tau : x_i^1 \in \{0, 1\}$ and $x_i^2 \in \{0, 1\}$
C5. $\forall \tau_i \in \text{H1}(1/R(\Pi), 1/R(\Pi), \tau) : x_i^1 = 1$
C6. $\forall \tau_i \in \text{H2}(1/R(\Pi), 1/R(\Pi), \tau) : x_i^2 = 1$
C7. $\text{H12}(1/R(\Pi), 1/R(\Pi), \tau) = \emptyset$
C8. $\sum_{\tau_i \in \tau}(C_i^1/T_i) \times x_i^1 \leq m_1 \times (s/R(\Pi))$
C9. $\sum_{\tau_i \in \tau}(C_i^2/T_i) \times x_i^2 \leq m_2 \times (s/R(\Pi))$

Fig. 10: Constraints expressing a necessary intra-migrative feasibility condition; rewritten.

C1. $\forall t \in TS(\tau, 1/R(\Pi)) : \left(\sum_{\tau_i \in \tau} \text{ffdbf}^{*1}(\tau_i, t, 1/R(\Pi), s, \tau) \times x_i^1\right) \leq m_1 \times t \times (s/R(\Pi)) \times 2$
C2. $\forall t \in TS(\tau, 1/R(\Pi)) : \left(\sum_{\tau_i \in \tau} \text{ffdbf}^{*2}(\tau_i, t, 1/R(\Pi), s, \tau) \times x_i^2\right) \leq m_2 \times t \times (s/R(\Pi)) \times 2$
C3. $\forall \tau_i \in \tau : x_i^1 + x_i^2 = 1$
C4. $\forall \tau_i \in \tau : x_i^1 \in \{0, 1\}$ and $x_i^2 \in \{0, 1\}$
C5. $\forall \tau_i \in \text{H1}(1/R(\Pi), 1/R(\Pi), \tau) : x_i^1 = 1$
C6. $\forall \tau_i \in \text{H2}(1/R(\Pi), 1/R(\Pi), \tau) : x_i^2 = 1$
C7. $\text{H12}(1/R(\Pi), 1/R(\Pi), \tau) = \emptyset$
C8. $\sum_{\tau_i \in \tau}(C_i^1/T_i) \times x_i^1 \leq m_1 \times (s/R(\Pi))$
C9. $\sum_{\tau_i \in \tau}(C_i^2/T_i) \times x_i^2 \leq m_2 \times (s/R(\Pi))$

Fig. 11: Constraints expressing a necessary intra-migrative feasibility condition; rewritten further.

C1. $\forall t \in TS(\tau, 1/R(\Pi)) : \left(\sum_{\tau_i \in \tau} \text{ffdbf}^{*1}(\tau_i, t, 1/R(\Pi), s, \tau) \times x_i^1\right) \leq (m_1 - (m_1 - 1) \times 1/R(\Pi)) \times t \times s \times 1/2$
C2. $\forall t \in TS(\tau, 1/R(\Pi)) : \left(\sum_{\tau_i \in \tau} \text{ffdbf}^{*2}(\tau_i, t, 1/R(\Pi), s, \tau) \times x_i^2\right) \leq (m_2 - (m_2 - 1) \times 1/R(\Pi)) \times t \times s \times 1/2$
C3. $\forall \tau_i \in \tau : x_i^1 + x_i^2 = 1$
C4. $\forall \tau_i \in \tau : x_i^1 \in \{0, 1\}$ and $x_i^2 \in \{0, 1\}$
C5. $\forall \tau_i \in \text{H1}(1/R(\Pi), 1/R(\Pi), \tau) : x_i^1 = 1$
C6. $\forall \tau_i \in \text{H2}(1/R(\Pi), 1/R(\Pi), \tau) : x_i^2 = 1$
C7. $\text{H12}(1/R(\Pi), 1/R(\Pi), \tau) = \emptyset$
C8. $\sum_{\tau_i \in \tau}(C_i^1/T_i) \times x_i^1 \leq ((m_1 - (m_1 - 1) \times 1/R(\Pi)) \times s \times 1/2)$
C9. $\sum_{\tau_i \in \tau}(C_i^2/T_i) \times x_i^2 \leq ((m_2 - (m_2 - 1) \times 1/R(\Pi)) \times s \times 1/2)$

Fig. 12: Constraints expressing a necessary intra-migrative feasibility condition; rewritten even more.

C1. $\forall t \in TS(\tau, 1/R(\Pi)) : \left(\sum_{\tau_i \in \tau} \text{ffdbf}^{*1}(\tau_i, t, 1/R(\Pi), s, \tau) \times x_i^1\right) \leq (m_1 - (m_1 - 1) \times 1/R(\Pi)) \times t \times s \times 1/2$
C2. $\forall t \in TS(\tau, 1/R(\Pi)) : \left(\sum_{\tau_i \in \tau} \text{ffdbf}^{*2}(\tau_i, t, 1/R(\Pi), s, \tau) \times x_i^2\right) \leq (m_2 - (m_2 - 1) \times 1/R(\Pi)) \times t \times s \times 1/2$
C3. $\forall \tau_i \in \tau : x_i^1 + x_i^2 = 1$
C4. $\forall \tau_i \in \tau : x_i^1 \geq 0$ and $x_i^2 \geq 0$
C5. $\forall \tau_i \in \text{H1}(1/R(\Pi), 1/R(\Pi), \tau) : x_i^1 = 1$
C6. $\forall \tau_i \in \text{H2}(1/R(\Pi), 1/R(\Pi), \tau) : x_i^2 = 1$
C7. $\text{H12}(1/R(\Pi), 1/R(\Pi), \tau) = \emptyset$
C8. $\sum_{\tau_i \in \tau}(C_i^1/T_i) \times x_i^1 \leq ((m_1 - (m_1 - 1) \times 1/R(\Pi)) \times s \times 1/2)$
C9. $\sum_{\tau_i \in \tau}(C_i^2/T_i) \times x_i^2 \leq ((m_2 - (m_2 - 1) \times 1/R(\Pi)) \times s \times 1/2)$

Fig. 13: Constraints expressing a necessary intra-migrative feasibility condition; rewritten to LP.

being integers (follows from the fact that it does not contain the tasks in F), it follows that X' also satisfies the following

constraints: Fig. 8 where in the expression on the right-hand side of C1,C2,C8,C9, the symbol s is replaced by $s/2$. Note that $F \subseteq \tau$ and hence from Eq. (23), it follows that for F, there is an X that satisfies the following constraints: Fig. 8 where in the expression on the right-hand side of C1,C2,C8,C9, the symbol s is replaced by $s/2$. Adding X' and X gives us a new matrix that satisfies Fig. 8 and this yields that $\mathrm{found}\mathrm{PTMILP}$ is true. This is a contradiction.

It can be seen that if the lemma is false then each case results in contradiction. Hence, the lemma is true. ■

Theorem 4. *Consider a taskset τ and a computer platform Π.* **If** *τ is intra-migrative feasible on $\Pi \times 1/R(\Pi)$* **then** *Algorithm 2 declares SUCCESS.*

Proof: Follows from Lemma 18 and Lemma 19. ■

Hence, the speedup factor of Algorithm 2 is $R(\Pi)$.

V. Conclusions

The problem of scheduling real-time tasks on a heterogeneous multiprocessor has received increasing attention from researchers during recent years but no solution was available for parallel tasks with proven speedup factor. Therefore, in this paper, we have presented the first algorithm for scheduling parallel tasks on a heterogeneous multiprocessor with proven speedup factor. We did so by focusing on constrained-deadline sporadic tasks and a heterogeneous multiprocessor where processors are of two types and we presented a new algorithm that assigns tasks to processor types and then apply global-Earliest-Deadline-First on each type of processors. Our new algorithm has pseudo-polynomial time complexity and speedup factor at most 5.

Acknowledgment

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8721-05-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center. This material has been approved for public release and unlimited distribution. DM-0002064

References

[1] G. Raravi, B. Andersson, V. Nélis, and K. Bletsas, "Task assignment algorithms for two-type heterogeneous multiprocessors," *Real-Time Systems*, 2014.

[2] S. Baruah, "Task partitioning upon heterogeneous multiprocessor platforms," in *RTAS*, 2004.

[3] G. Raravi and V. Nélis, "Task assignment algorithms for heterogeneous multiprocessors," *ACM TECS*, 2014.

[4] S. Baruah, "Partitioning real-time tasks among heterogeneous multiprocessors," in *ICPP*, 2004.

[5] G. Raravi, B. Andersson, and K. Bletsas, "Assigning real-time tasks on heterogeneous multiprocessors with two unrelated types of processors," *Real-Time Systems*, 2013.

[6] G. Raravi and V. Nélis, "A PTAS for assigning sporadic tasks on two-type heterogeneous multiprocessors," in *RTSS*, 2012.

[7] B. Andersson and G. Raravi, "Provably good task assignment for two-type heterogeneous multiprocessors using cutting planes," *ACM TECS*, 2014.

[8] S. Kedad-Sidhoum, F. Monna, G. Mouni, and D. Trystram, "Scheduling independent tasks on multi-cores with GPU accelerators," in *Euro-Par*, ser. Springer Lecture Notes in Computer Science, 2014.

[9] A. Wiese, V. Bonifaci, and S. Baruah, "Partitioned EDF scheduling on a few types of unrelated multiprocessors," *Real-Time Systems*, 2013.

[10] J. Lenstra, D. Shmoys, and E. Tardos, "Approximation algorithms for scheduling unrelated parallel machines," *Math. Program.*, 1990.

[11] J. Correa, M. Skutella, and J. Verschae, "The power of preemption on unrelated machines and applications to scheduling orders," *Mathematics of Operations Research*, 2012.

[12] E. Horowitz and S. Sahni, "Exact and Approximate Algorithms for Scheduling Nonidentical Processors," *Journal of the ACM*, 1976.

[13] K. Jansen and L. Porkolab, "Improved approximation schemes for scheduling unrelated parallel machines," in *STOC*, 1999.

[14] M. Niemeier, A. Wiese, and S. Baruah, "Partitioned real-time scheduling on heterogeneous shared-memory multiprocessors," in *ECRTS*, 2011.

[15] A. Marchetti-Spaccamela, C. Rutten, S. van der Ster, and A. Wiese, "Assigning sporadic tasks to unrelated parallel machines," in *ICALP*, 2012.

[16] K. Lakshmanan, S. Kato, and R. Rajkumar, "Scheduling parallel real-time tasks on multi-core processors," in *RTSS*, 2010.

[17] A. Saifullah, K. Agrawal, C. Lu, and C. Gill, "Multi-core real-time scheduling for generalized parallel task models," in *RTSS*, 2011.

[18] B. Andersson and D. de Niz, "Analyzing Global-EDF for multiprocessor scheduling of parallel tasks," in *OPODIS*, 2012.

[19] V. Bonifaci, A. Marchetti-Spaccamela, S. Stiller, and A. Wiese, "Feasibility analysis in the sporadic DAG model," in *ECRTS*, 2013.

[20] S. Baruah, V. Bonifaci, A. Marchetti-Spaccamela, L. Stougie, and A. Wiese, "A generalized parallel task model for recurrent real-time processes," in *RTSS*, 2012.

[21] J. Kim, H. Kim, K. Lakshmanan, and R. R. Rajkumar, "Parallel scheduling for cyber physical systems: Analysis and case study on a self-driving car," in *RTAS*, 2013.

[22] J. Li, K. Agrawal, C. Lu, and C. Gill, "Analysis of global EDF for parallel tasks," in *ECRTS*, 2013.

[23] L. Cong and J. H. Anderson, "Supporting soft real-time DAG-based systems on multiprocessors with no utilization loss," in *RTSS*, 2010.

[24] S. Kato and Y. Ishikawa, "Gang EDF scheduling of parallel task systems," in *RTSS*, 2009.

[25] P. Jayachandran and T. Abdelzaher, "Reduction-based schedulability analysis of distributed systems with cycles in the task graph," *Real-Time Systems*, 2010.

[26] M. Qamhieh, F. Fauberteau, G. Laurent, and S. Midonnet, "Global EDF scheduling of directed acyclic graphs on multiprocessor systems," in *RTNS*, 2013.

[27] D. Ferry, J. Li, M. Mahadevan, K. Agrawal, C. Gill, and C. Lu, "A real-time scheduling service for parallel tasks," in *RTAS*, 2013.

[28] H. S. Chwa, J. Lee, K.-M. Phan, A. Easwaran, and I. Shin, "Global EDF schedulability analysis for synchronous parallel tasks on multicore platforms," in *ECRTS*, 2013.

[29] S. Collette, L. Cucu, and J. Goossens, "Integrating job parallelism in real-time scheduling theory," *Information Processing Letters*, 2008.

[30] L. Nogueira and L. P. Pinho, "Server-based scheduling of parallel real-time tasks," in *EMSOFT*, 2012.

[31] P. Axer, S. Quinton, M. Neukirchner, R. Ernest, B. Döbel, and H. Härtig, "Response-time analysis of parallel fork-join workloads with real-time constraints," in *ECRTS*, 2013.

[32] Q. Wang and G. Parmer, "FJOS: Practical, predictable, and efficient system support for fork-join parallelism," in *RTAS*, 2014.

[33] P. Fauberteau, S. Midonnet, and M. Qamhieh, "Partitioned scheduling of parallel real-time tasks on multiprocessor systems," *SIGBED Review*, 2011.

[34] M. Holenderski, R. J. Bril, and J. J. Lukkien, "Parallel-task scheduling on multiple resources," in *ECRTS*, 2012.

[35] N. Fisher, J. Goossens, and S. K. Baruah, "Optimal online multiprocessor scheduling of sporadic real-time tasks is impossible," *Real-Time Systems*, 2010.

[36] C. A. Phillips, C. Stein, E. Torng, and J. Wein, "Optimal time-critical scheduling via resource augmentation," in *STOC*, 1997.

[37] T. P. Baker and M. Cirinei, "Brute-force determination of multiprocessor schedulability for sets of sporadic hard-deadline tasks," in *OPODIS*, 2007.

[38] T. P. Baker, "Multiprocessor EDF and deadline monotonic schedulability analysis," in *RTSS*, 2003.

[39] M. Bertogna, M. Cirinei, and G. Lipari, "Improved schedulability analysis of EDF on multiprocessor platforms," in *ECRTS*, 2005.

[40] S. K. Baruah, V. Bonifaci, A. Marchetti-Spaccamela, and S. Stiller, "Implementation of a speedup-optimal global EDF schedulability test," in *ECRTS*, 2009.
[41] N. Karmarkar, "A new polynomial time algorithm for linear programming," *Combinatorica*, 1984.

APPENDIX

A. *Proof of Lemma 4*

In this section, we prove Lemma 4 and we do this incrementally, i.e., by proving some basic results and then merging them to obtain the desired result.

Lemma 20. $\forall t > \text{mi}(\tau) : \text{ffdbf}(\tau_i, t, v, s) \leq \text{ffdbf}(\tau_i, \text{mi}(\tau), v, s) + \left(C_i + \frac{C_i}{T_i} \times (t - \text{mi}(\tau))\right)$

Proof: Algebraic manipulations yield:

$$\begin{aligned} t &= \text{mi}(\tau) + (t - \text{mi}(\tau)) \\ &= \text{mi}(\tau) + \left\lfloor \frac{t - \text{mi}(\tau)}{T_i} \right\rfloor \times T_i + (t - \text{mi}(\tau)) \mod T_i \\ &\leq \text{mi}(\tau) + \left\lfloor \frac{t - \text{mi}(\tau)}{T_i} \right\rfloor \times T_i + T_i \\ &\leq \text{mi}(\tau) + \left(\left\lfloor \frac{t - \text{mi}(\tau)}{T_i} \right\rfloor + 1\right) \times T_i \end{aligned}$$

Using this on Lemma 1 yields:

$$\text{ffdbf}(\tau_i, t, v, s) \leq \text{ffdbf}\left(\tau_i, \text{mi}(\tau) + \left(\left\lfloor \frac{t - \text{mi}(\tau)}{T_i} \right\rfloor + 1\right) \times T_i, v, s\right)$$

Using Lemma 2 yields:

$$\text{ffdbf}(\tau_i, t, v, s) \leq \text{ffdbf}(\tau_i, \text{mi}(\tau), v, s) + \left(\left\lfloor \frac{t - \text{mi}(\tau)}{T_i} \right\rfloor + 1\right) \times C_i$$

Relaxing the bound on the right-hand side and rewriting yields:

$$\text{ffdbf}(\tau_i, t, v, s) \leq \text{ffdbf}(\tau_i, \text{mi}(\tau), v, s) + (t - \text{mi}(\tau)) \times \frac{C_i}{T_i} + C_i$$

This states the lemma. ■

Lemma 21. $\forall t > 0, t \leq \text{mi}(\tau) : \text{ffdbf}(\tau_i, t, v, s) \leq \text{ffdbf}\left(\tau_i, 2^{\lfloor \log_2 t \rfloor + 1}, v, s\right)$

Proof: Follows from Lemma 1 (monotonicity) and observing that $t \leq 2^{\lfloor \log_2 t \rfloor + 1}$. ■

We now restate Lemma 4 and prove it.

Lemma 4. $\text{ffdbf}(\tau_i, t, v, s) \leq \text{ffdbf}^*(\tau_i, t, v, s, \tau)$

Proof: We need to consider two cases.

Case 1. $t > \text{mi}(\tau)$: For this case Lemma 20 along with the definition of ffdbf^* in Eq. (9) proves the lemma.

Case 2. $t \leq \text{mi}(\tau)$: For this case Lemma 21 along with the definition of ffdbf^* in Eq. (9) proves the lemma. ■

B. *Proof of Lemma 14 and Lemma 15*

In this section, we prove Lemma 14 and Lemma 15 and once again we do this incrementally, i.e., by proving some basic results and then merging them to obtain the desired result.

Lemma 22. $\forall t \in (\text{mi}(\tau), \text{mi}(\tau) + T_i] : \text{ffdbf}^*(\tau_i, t, v, s, \tau) \leq \text{ffdbf}(\tau_i, t, v, s) + 2 \times C_i$

Proof: Since $t > \text{mi}(\tau)$ and because of Lemma 1 (monotonicity), we have:

$$\text{ffdbf}(\tau_i, \text{mi}(\tau), v, s) \leq \text{ffdbf}(\tau_i, t, v, s) \tag{27}$$

Rewriting yields:

$$\text{ffdbf}\left(\tau_i, \text{mi}(\tau), v, s\right) + C_i + \frac{C_i}{T_i} \times T_i \leq \text{ffdbf}\left(\tau_i, t, v, s\right) + 2 \times C_i \quad (28)$$

From $t \in (\text{mi}(\tau), \text{mi}(\tau) + T_i]$, we obtain that: $t - \text{mi}(\tau) \leq T_i$. Applying this on Eq. (28) yields: $\forall t \in (\text{mi}(\tau), \text{mi}(\tau) + T_i]$:

$$\text{ffdbf}\left(\tau_i, \text{mi}(\tau), v, s\right) + C_i + \frac{C_i}{T_i} \times (t - \text{mi}(\tau)) \leq \text{ffdbf}\left(\tau_i, t, v, s\right) + 2 \times C_i \quad (29)$$

Using the definition of ffdbf^* yields: $\forall t \in (\text{mi}(\tau), \text{mi}(\tau) + T_i]$:

$$\text{ffdbf}^*\left(\tau_i, t, v, s, \tau\right) \leq \text{ffdbf}\left(\tau_i, t, v, s\right) + 2 \times C_i$$

Hence the proof. ∎

Lemma 23. $\forall t > \text{mi}(\tau) : \text{ffdbf}^*\left(\tau_i, t, v, s, \tau\right) \leq \text{ffdbf}\left(\tau_i, t, v, s\right) + 2 \times C_i$

Proof: We prove this by contradiction. If the lemma was false then there exist a t such that $t > \text{mi}(\tau)$ and

$$\text{ffdbf}^*\left(\tau_i, t, v, s, \tau\right) > \text{ffdbf}\left(\tau_i, t, v, s\right) + 2 \times C_i$$

If $t > \text{mi}(\tau) + T_i$ then decreasing t by T_i decreases the left-hand side and the right-hand side of the above inequality by the same amount (C_i). Hence, we can decrease t by T_i until it holds that $t \in (\text{mi}(\tau), \text{mi}(\tau) + T_i]$. And this gives us that there exists a t such that $t \in (\text{mi}(\tau), \text{mi}(\tau) + T_i]$ and

$$\text{ffdbf}^*\left(\tau_i, t, v, s, \tau\right) > \text{ffdbf}\left(\tau_i, t, v, s\right) + 2 \times C_i$$

But this contradicts Lemma 22 and hence it is not possible that lemma under discussion is false. Hence the proof. ∎

Lemma 24. $\forall t > \text{mi}(\tau) : \text{ffdbf}^*\left(\tau_i, t, v, s, \tau\right) \leq \text{ffdbf}\left(\tau_i, t, v, s\right) \times 2$

Proof: Since $t > \text{mi}(\tau)$, it follows that $t > \text{TMAX} + \text{DMAX}$ and then it follows that:

$$\text{ffdbf}\left(\tau_i, t, v, s\right) \geq 2 \times C_i \quad (30)$$

Using Lemma 23 and Eq. (30) yields: $\forall t > \text{mi}(\tau)$:

$$\begin{aligned} \frac{\text{ffdbf}^*\left(\tau_i, t, v, s, \tau\right)}{\text{ffdbf}\left(\tau_i, t, v, s\right)} &\leq \frac{\text{ffdbf}\left(\tau_i, t, v, s\right) + 2 \times C_i}{\text{ffdbf}\left(\tau_i, t, v, s\right)} \\ &\leq 1 + \frac{2 \times C_i}{\text{ffdbf}\left(\tau_i, t, v, s\right)} \\ &\leq 1 + \frac{2 \times C_i}{2 \times C_i} \\ &\leq 2 \end{aligned} \quad (31)$$

Rewriting yields: $\forall t > \text{mi}(\tau)$:

$$\text{ffdbf}^*\left(\tau_i, t, v, s, \tau\right) \leq \text{ffdbf}\left(\tau_i, t, v, s\right) \times 2$$

This states the lemma. ∎

Lemma 14. $\forall Q \geq 0, \forall v \geq 0, \forall s \geq 0$:
$(\forall t \geq 0 : \left(\sum_{\tau_i \in \tau} \text{ffdbf}\left(\tau_i, t, v, s\right)\right) \leq m \times t \times Q)$
$\Rightarrow$
$(\forall t \in TS(\tau, \theta) : \left(\sum_{\tau_i \in \tau} \text{ffdbf}^*\left(\tau_i, t, v, s, \tau\right)\right) \leq m \times t \times Q \times 2)$

Proof: Suppose that the lemma was false. Then it holds that there exists a Q, v and s such that $Q \geq 0, v \geq 0, s \geq 0$ and

$$\left(\forall t \geq 0 : \left(\sum_{\tau_i \in \tau} \text{ffdbf}\left(\tau_i, t, v, s\right)\right) \leq m \times t \times Q\right) \wedge \left(\exists t \in TS(\tau, \theta) : \left(\sum_{\tau_i \in \tau} \text{ffdbf}^*\left(\tau_i, t, v, s, \tau\right)\right) > m \times t \times Q \times 2\right)$$

Since there exists a t in $TS(\tau, \theta)$ such that the last constraint is true, let us choose one of them and call it t_0. This gives us:

$$\left(\forall t \geq 0 : \left(\sum_{\tau_i \in \tau} \text{ffdbf}\left(\tau_i, t, v, s\right)\right) \leq m \times t \times Q\right) \wedge \left(\left(\sum_{\tau_i \in \tau} \text{ffdbf}^*\left(\tau_i, t_0, v, s, \tau\right)\right) > m \times t_0 \times Q \times 2\right) \quad (32)$$

Let us consider two cases:

Case 1: $t_0 > \text{mi}(\tau)$. Applying Lemma 24 on the last constraint in Eq. (32) yields:

$$\left(\forall t \text{ s.t. } t \geq 0 : \left(\sum_{\tau_i \in \tau} \text{ffdbf}\left(\tau_i, t, v, s\right)\right) \leq m \times t \times Q\right) \wedge \left(\sum_{\tau_i \in \tau} \text{ffdbf}\left(\tau_i, t_0, v, s\right) \times 2 > m \times t_0 \times Q \times 2\right)$$

Dividing the last constraint by 2 yields:

$$\left(\forall t \geq 0 : \left(\sum_{\tau_i \in \tau} \text{ffdbf}\left(\tau_i, t, v, s\right)\right) \leq m \times t \times Q\right) \wedge \left(\sum_{\tau_i \in \tau} \text{ffdbf}\left(\tau_i, t_0, v, s\right) > m \times t_0 \times Q\right)$$

This is a contradiction. End of Case 1.

Case 2: $t_0 \leq \text{mi}(\tau)$. Applying $2 \times t_0$ on the 1st constraint in Eq. (32) and relaxing the last constraint yields:

$$\left(\left(\sum_{\tau_i \in \tau} \text{ffdbf}\left(\tau_i, 2 \times t_0, v, s\right)\right) \leq m \times 2 \times t_0 \times Q\right) \wedge \left(\left(\sum_{\tau_i \in \tau} \text{ffdbf}^*\left(\tau_i, t_0, v, s, \tau\right)\right) > m \times t_0 \times Q \times 2\right)$$

Since $t_0 \in TS(\tau, \theta)$, it follows from the definition of ffdbf^* that $\text{ffdbf}^*\left(\tau_i, t_0, v, s, \tau\right) = \text{ffdbf}\left(\tau_i, 2 \times t_0, v, s\right)$. Applying

this on the last constraint yields:

$$\left(\left(\sum_{\tau_i \in \tau} \text{ffdbf}\left(\tau_i, 2 \times t_0, v, s\right)\right) \leq m \times 2 \times t_0 \times Q\right)$$
$$\wedge$$
$$\left(\left(\sum_{\tau_i \in \tau} \text{ffdbf}\left(\tau_i, 2 \times t_0, v, s\right)\right) > m \times t_0 \times Q \times 2\right)$$

This is a contradiction. End of Case 2.

It can be seen that if the lemma is false then for each case, we obtain a contradiction. Hence, the lemma is true. ■

Lemma 15. $\forall Q \geq 0, \forall s \geq 0:$
$(\forall t \geq 0 : \left(\sum_{\tau_i \in \tau} \text{ffdbf}^2\left(\tau_i, t, \theta_2, s\right)\right) \leq m \times t \times Q)$
$\Rightarrow$
$(\forall t \in TS(\tau, \theta_2) : \left(\sum_{\tau_i \in \tau} \text{ffdbf}^{*2}\left(\tau_i, t, \theta_2, s\right)\right) \leq m \times t \times Q \times 2)$

Proof: Analogous to the proof of Lemma 14. ■